MATH 557 Oct 20

Computable functions

Turing machines

A **Turing machine** M has an unlimited (both sides) **tape** which is partitioned into infinitely many **cells**. Each cell can hold a letter from a finite **alphabet** $s_1, \dots, s_n$. At any time, only finitely many cells contain a symbol $\in \{s_1, \dots, s_n\}$. The empty cells are marked by a *blank* symbol b that is distinct from the s_i. To facilitate things, we put $s_0 := b$.

Access to the tape is given via a **read-write head**, which scans a single cell. The cell currently scanned can be read and its contents overwritten. The head can also move left or right. These actions can be made dependent on the contents of the scanned cell.

Thus, the machine can read any cell on the tape and read and write its contents. Furthermore, at any time M is in one of finitely many **states** $q_1, \dots, q_r$. The states influence the behavior of M.

Formally, there are three types of operations (also called *instructions*): Suppose M is in state q_i and currently scan a cell with symbol s_j $(0 \leq j \leq n)$.

1. $q_i\ s_j\ s_k\ q_l$: delete s_j write s_k,
2. $q_i\ s_j\ R\ q_l$: move the head one cell to the right,
3. $q_i\ s_j\ L\ q_l$: move the head one cell to the left.

In each case, transition to q_l afterwards.

A (deterministic) **Turing program** is a finite list of instructions (1)-(3) such that for any pair (q_i, s_k) there is at most one instruction starting with q_i, s_k.

The computation of a Turing program

The machine begins in state q_1 scanning the leftmost cell not containing a blank and follows the instructions, describe above, according to the current state and the content of the current cell.

If the machine enters state q_r, the computation **halts** (no further instructions will be applied). If it ever enters a state q_i and scans symbol s_j for which there is no applicable instruction, it **stalls** (in particular, there will be no output as defined below).

During the computation, the machine will pass from one **configuration** to the next. A configuration consists of

> the current state + the current cell scanned + the contents the left and right of the current cell up to the leftmost/rightmost non-blank symbol.

Every configuration is determined by a finite amount of information.

An example

The alphabet of M consists of the symbols $0, 1$ (and b for a blank cell), the states are q_1, q_2, q_3. The program P has the following instructions:

$$q_1\ 0\ R\ q_1$$
$$q_1\ 1\ 0\ q_2$$
$$q_2\ 0\ R\ q_2$$
$$q_2\ 1\ R\ q_1$$
$$q_2\ b\ b\ q_3$$

In the initial configuration, the machine looks as follows:

				↓									
…	b	b	b	1	1	1	1	1	1	b	b	b	…

M follows the instructions above until it reaches the halting state q_3

										↓			
…	b	b	b	0	1	0	1	0	1	b	b	b	…

Turing-computable functions

Since Turing computations may not halt (either because they stall or enter an infinite loop and never reach the halting state), we define **partial computable functions** whose domain may be a proper subset of $\mathbb{N}^n$.

We use the following notation:

$$\begin{aligned} f(\vec{x})\downarrow &: \iff f \text{ is defined for } \vec{x} \\ f(\vec{x})\uparrow &: \iff f \text{ is not defined for } \vec{x} \\ f(\vec{x}) \simeq g(\vec{x}) &: \iff f(\vec{x})\downarrow \iff g(\vec{x})\downarrow \quad \text{and} \quad f(\vec{x})\downarrow \implies f(\vec{x}) = g(\vec{x}). \end{aligned}$$

Definition 0.1.
We say a Turing machine M with program P **computes** a partial function $f :\subseteq \mathbb{N} \to \mathbb{N}$ if the machine, when starting with $n+1$ many 1's, ends in state q_r if and only if n is in the domain of f, and in this case, when M reaches state q_r, there are exactly $f(n)$ many 1's on the tape (not necessarily contiguous).

A partial function f is **Turing computable** if and only if there exists a Turing machine M and a program P that computes it.

The definition for multidimensional functions is similar.